

PR #39636 完整报告

sgl-project/sglang

[Test] Fix DeepGEMM batch invariance test output dtype

合并时间: 2026-09-16 03:40

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39636>

执行摘要

- 一句话: 修复 DeepGEMM batch 不变测试缺 `out_dtype` 参数所致 nightly 失败
- 推荐动作: 这是一个小而规范的单点修复, 值得快速合入但无需精读。值得关注的设计点是: 签名契约变更 (#38176) 时未同步扫描直接调用方, 暴露了“库函数签名变更缺乏调用点检索”的过程问题; 建议团队在后续类似 API 收紧时用 `grep` 或编译期检查覆盖测试直接调用。

功能与动机

PR #38176 为确定性推理把 router GEMM 保持在 fp32, 同时令 `_matmul_persistent_deepgemm` 的 `out_dtype` 变为必填参数, 但测试 `test_deepgemm_random_bf16_batch_invariance` 中的直接调用仍只传两个参数, 导致 main nightly 任务报 `TypeError: _matmul_persistent_deepgemm() missing 1 required positional argument: 'out_dtype'`。作者在 PR body 中明确说明这是测试调用点未跟上内核 API 契约变化。

实现拆解

1. 定位破坏源: PR #38176 在 `python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py` 中把 `_matmul_persistent_deepgemm` 的 `out_dtype` 改为必填参数, 而 `test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py` 中 `test_deepgemm_random_bf16_batch_invariance` 的两处直调未适配。
2. 修复调用点: 在 reference 调用 `_matmul_persistent_deepgemm(a[:1], b)` 和循环内 `batched` 调用 `_matmul_persistent_deepgemm(a[:batch_size], b)` 各补上 `out_dtype=torch.bfloat16`, 与输入张量 `a`、`b` 的 BF16 dtype 保持一致, 维持原测试语义。
3. 验证配套: 作者做了 Python 语法编译检查、绑定两处调用点与 helper 当前签名 (改动前 fail、改动后 pass)、`git diff --check`; 随后通过 `/rerun-test` 在 1-gpu-h100 上重跑该用例, bot 反馈测试通过。无新增测试、配置或部署改动。

关键文件:

- `test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py` (模块 批量不变; 类别 test; 类型 test-coverage; 符号 `TestBatchInvariantOps.test_deepgemm_random_bf16_batch_invariance`): 唯一变更文件, 修正 `test_deepgemm_random_bf16_batch_invariance` 中两处对 `_matmul_persistent_deepgemm` 的直接调用, 补齐 #38176 引入的必

填参数 out_dtype, 恢复 nightly 测试稳定性。

关键符号: TestBatchInvariantOps.test_deepgemm_random_bf16_batch_invariance

关键源码片段

test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py

唯一变更文件, 修正 test_deepgemm_random_bf16_batch_invariance 中两处对 _matmul_persistent_deepgemm 的直接调用, 补齐 #38176 引入的必填参数 out_dtype, 恢复 nightly 测试稳定性。

```
# test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py
# 本用例验证 DeepGEMM 在 batch-invariant 模式下, 不同 batch size 的输出与单行 reference
# 完全一致。
# #38176 收紧 API 后 out_dtype 变为必填参数, 此处两处直调均需显式传入。
def test_deepgemm_random_bf16_batch_invariance(self):
    if not batch_invariant_ops.ENABLE_JIT_DEEPGEMM:
        self.skipTest("DeepGEMM is unavailable on this device")

    deep_gemm = batch_invariant_ops.deep_gemm
    get_deterministic = getattr(deep_gemm, "get_deterministic_algorithms", None)
    original_mode = get_deterministic() if get_deterministic else None
    modes = (False, True) if get_deterministic else (None,)
    generator = torch.Generator(device=device_type).manual_seed(42)
    # 输入 a、b 均为 bfloat16, 因此 out_dtype 必须显式传 torch.bfloat16
    # 才能保持原测试语义, 否则调用会因缺少必填参数而抛出 TypeError。
    a = torch.randn(257, 4096, dtype=torch.bfloat16, generator=generator)
    b = torch.randn(4096, 4096, dtype=torch.bfloat16, generator=generator).T
    try:
        for mode in modes:
            if mode is not None:
                deep_gemm.use_deterministic_algorithms(mode)
            with self.subTest(deterministic=mode):
                # reference 用 batch 大小为 1 的结果, 作为 batch-invariant 的基准。
                ref = batch_invariant_ops._matmul_persistent_deepgemm(
                    a[:1], b, out_dtype=torch.bfloat16
                )
                for batch_size in (16, 64, 257):
                    # batched 调用同样补上 out_dtype, 与 reference 保持同一输出 dtype。
                    out = batch_invariant_ops._matmul_persistent_deepgemm(
                        a[:batch_size], b, out_dtype=torch.bfloat16
                    )
                    torch.testing.assert_close(out[:1], ref, rtol=0, atol=0)
                if get_deterministic:
                    self.assertEqual(get_deterministic(), mode)
    finally:
        if original_mode is not None:
            deep_gemm.use_deterministic_algorithms(original_mode)
```

评论区精华

评论区内没有正式 review 线程，核心交流围绕回归验证与破坏来源确认：

- YAMY1234 发起 /rerun-test, github-actions[bot] 反馈 1-gpu-h100 上该测试  通过，完成回归验证。
- 维护者 b8zhong 评论 [Thanks @YAMY1234! Sorry about that](#), 承认这是 #38176 改动 API 契约导致的遗漏，确认破坏来源并认可测试侧修复即可。
- 修复后的回归验证 (testing): 修复在真实 DeepGEMM GPU 上完成回归验证，测试恢复绿色。
- 破坏来源确认 (other): 各方确认破坏来自 #38176 的 API 契约收紧，本 PR 以测试侧适配收尾，无需改动内核。

风险与影响

- 风险：改动仅涉及测试文件两个调用点，无运行时路径、内核或配置变更，回归风险极低。唯一值得注意的是该测试与 `_matmul_persistent_deepgemm` 的签名契约强耦合：未来若 `out_dtype` 语义或默认值再次变化，此调用点（及仓库内其他直调该 helper 的测试）可能再次失效，建议后续通过统一封装或参数默认值来缓解。另外测试依赖 DeepGEMM GPU 环境，非 H100/ 兼容卡上会走 `skipTest` 路径，不影响其他平台 CI。
- 影响：影响范围限于 `test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py` 单个用例，直接影响是恢复 `main nightly` 中 `deepgemm batch invariance` 测试的稳定性，消除因 API 契约变更导致的 `TypeError` 噪音。对用户无感知，对团队的价值在于保住 `batch-invariant` 确定性回归防线，防止 DeepSeek 系列模型路由层 `fp32` 确定性逻辑后续被无意破坏。
- 风险标记：仅测试调用点适配，无运行时影响，契约耦合于 #38176 内核 API

关联脉络

- PR #38176 keeping router GEMM in fp32 for deterministic inference (DeepSeek V3/V4): 本 PR 的直接诱因：该 PR 令 `_matmul_persistent_deepgemm` 的 `out_dtype` 成为必填参数，导致测试直调报 `TypeError`。
- PR #38160 [Feature] Support BF16 and batch-invariant inference with DeepEP v2: 同一 `batch-invariant` 特性线，涉及 `batch_invariant_ops` 与 DeepGEMM/DeepEP 内核的 BF16 确定性语义，本 PR 维护的正是这条回归防线。